Simulation Of A Polling System For Token Passing Networks Using GPSS

Edward Lo and R.H.S. Hardy

CSS/LCCR TR 89-07

92-05567/

TABLE OF CONTENTS

I. MOTIVATION 2

II. HARDWARE IMPLEMENTATION 3

III. ANALYTICAL MODEL 4
A. Analysis of a General Ring 5
B. Analysis of Identical Stations 5
C. Analysis of a Dual Latency Ring 6

IV. SIMULATION MODEL 7
A. Simulation Language 7
B. Simulation Models 8
1) Service Disciplines 9
2) Token-Operations 10
3) Avalanche Simulation 12
4) Turbo Mode 13
C. Statistical Considerations 14
1) Independent Replications of Runs 14
2) The Regenerative Method 15
D. Validation 16
E. Features of the Simulation Models 16

V. SIMULATION RESULTS 19

VI. CONCLUSION 21

VII. SUGGESTIONS FOR FUTURE WORK 22
A. Real-Time Applications 22
B. End-to-end Performance Modeling 22

REFERENCES 23

SIMULATION OF A POLLING SYSTEM FOR TOKEN PASSING NETWORKS USING GPSS*

Edward Lo, R.H.S. Hardy
School of Engineering Science
Simon Fraser University
Burnaby, B.C. V5A 1S6
Canada

Tel. (604) 291-4332

June, 1989

ABSTRACT

A token passing computer network uses a token to control access to the network and can be modeled as a polling system. The ANSI standard X3T9.5 and the IEEE standards 802.4 and 802.5 all use token passing as their medium access method. This paper presents a simulation model written in GPSS block statements that can simulate the performance of token passing networks. The performance is based on the number of stations, the traffic utilization, service disciplines, service time distributions, switchover time distributions and station latency distributions. The performance evaluation is very helpful for network design and planning.

---

*This work was supported by the Centre for Systems Science, Simon Fraser University and by the Natural Sciences and Engineering Research Council of Canada.

I. MOTIVATION

Local Area Networks (LANs) recently attain a lot of attention because of the needs of connections among different types of equipments and workstations and the progress in VLSI (Very Large Scale Integration) in microelectronics. This paper deals with the simulation aspects of the performance of token passing networks. The block diagrams of the simulation model are shown and are used as modules in GPSS.

The station latency, i.e. the time for data to past through a station, plays an important part in the performance of the token passing networks (see Figure 1). For example, Schwartz[1] recognized the influence of *walk-time* on the performance of polling systems. Bux[2] analyzed the effect of station latency on the message transfer delay of token rings and demonstrated the marked increase in delay produced by an increase in latency. More recently, recognition has been given to the effect of protocol design and of detailed station adapter implementation on the transmission delay performance of practical token passing networks.

In particular, we analyzed the effects on performance of a new method of selectively reducing station latency, the dual latency method[3,4]. It was proposed that the station latency be reduced by skipping the encoding and decoding operations in the network interface unit (NIU). A station can enter a lower latency state whenever its message queue is empty. The simulation results and analytical results were used to decide if the implementation of dual latency NIU was cost effective.

II. HARDWARE IMPLEMENTATION

There are two national standards of local area networks that are of particular interest. They are the IEEE (Institute of Electrical and Electronic Engineers) 802.5 (Token Ring), and the ANSI (American National Standards Institute) X3T9.5 (FDDI) standards. A token ring network consists of stations connected serially and information is transferred bit by bit from one station to the next as shown in Figure 1. The station which has the same address as the destination address copies information as it passes through the station. The information, finally, would be removed from the ring by the sending station. The right to transmit is controlled by a token being passed in the ring. Any station may capture the token by changing the free token to a busy token. The maximum amount of time that a station can use the ring before releasing a free token is controlled by a token holding timer set during ring initialization.

There are several conditions that must be met before a station can enter the low-latency state. Firstly, the station must not be sending a frame; otherwise, the data source is from the interface itself and not from the ring. Secondly, the station which has queued message(s) to send must decode the ring input to change a free token to a busy token on time. In the low-latency mode, a station acts simply as a repeater most of the time, i.e. it essentially repeats the data unaltered.

The complete network interface unit for the IEEE 802.5 standard was implemented using logic elements with timing parameters consistent with the specifications of a commercial 3 micron CMOS gate array process[3]. The entire circuit was simulated using a logic simulator at 10 Mbps. The results show that the full latency is 3.6 bits (360ns), whereas the latency is reduced to 0.6 bits (60ns) in low latency mode. In this case, a factor of 6 reduction in station latency was observed.

III. ANALYTICAL MODEL

A polling system consists of a system of multiple queues served by a single server in a cyclic order and a token ring is one type of polling systems. The most extensive book on this topic seems to be from Takagi[5]. Papers that are of interest can also be found[6,7,8,9]. The usual assumptions are as follows: Poisson arrivals, infinite buffer size, FCFS (First Come First Serve) for each message in a queue, multiple-token operation and a stable system. (For *multiple-token operation*, a transmitting station releases a new free token right after completion of transmission. For *single-token operation*, the station waits until its header returned before releasing a new free token. For *single-packet operation*, the station waits until all its data returned before releasing a new free token.) The time it takes to transmit a bit from one station to the next is called *switchover time*. The switchover time is equal to the station latency plus the propagation delay. The dual latency method tries to minimize the station latency while the propagation delay is fixed.

There are three main service disciplines that are different depending on the number of messages which may be served in a queue after a free token is captured; they are *exhaustive*, *gated* and *limited-to-one* (or *ordinary*) service disciplines. For exhaustive service, the server continues to serve the station until the queue is empty. For gated service, the server serves only those messages which are queued at the polling instance. For limited-to-1 service, the server serves at most one message. In the exhaustive or gated service system, if there is one heavily loaded station, the whole system will be overloaded. In the limited-to-one system, however, one heavily loaded station will affect the mean waiting time of that station dramatically rather than that of the whole system. This localization effect provides fairness among stations.

A. *Analysis of a General Ring*

For a general ring, numerical algorithms[7] are available to enable the calculation of mean waiting times in a gated or exhaustive token ring system. In an ordinary service system, the exact solution for the mean waiting time is not yet known. Only the two-queue case for a general ordinary service ring system has been solved explicitly. Therefore, approximation techniques[8] are available, but they are not incorporated because their accuracy is not guaranteed, especially for a comparison between a full latency and dual latency system.

B. *Analysis of Identical Stations*

Although the general exact solutions of token rings with ordinary service are not available, there is one special case of token rings that is well analyzed. It is the case of identical stations[5]. In this case, every station is assumed statistically identical in terms of its arrival rate, service time distribution, and switchover time distribution. The mean waiting time for limited-to-one service is given by:

$$E[w]\,|_{\text{limited-to-one}} = \frac{\delta^2}{2r} + \frac{N[\lambda b^{(2)} + r(1+\lambda b) + \lambda\delta^2]}{2[1 - N\lambda(r+b)]}$$

where $E[w]$ is the mean waiting time of messages queued for transmission at a station, r and δ^2 are the mean and variance of the switchover time (the sum of station latency and interstation propagation delay), N is the number of stations connected to the network, λ is the Poisson arrival rate of each station, b and $b^{(2)}$ are the first and second moments of the message service time respectively.

C. *Analysis of a Dual Latency Ring*

The evaluation of polling system performance under conditions of dependent statistical distributions is difficult in general. Ferguson[9] has developed bounds and approximations to the mean waiting time for a token ring with station dependent overheads and exhaustive service. For the case of limited-to-one service, no such approximate solutions are available yet. However, in the case of a polling system having a switchover time with two values, dependent on the state of the message queue, a relatively straightforward approach is possible for identical stations. We could approximate the solution by first assigning each station a fixed switchover time, equal to the sum of the low station latency value and the interstation propagation delay. We then add the additional latency, the difference between the full and low latency values, to the message service time.

IV. SIMULATION MODEL

Why Simulate ?

There are two general reasons for using a computer simulation model. Firstly, an exact mathematical analysis for a token ring with limited-to-one service is not yet possible as in the case here. Although approximation techniques are available for certain situations, the accuracy is not guaranteed. Hence, it may be inappropriate to compare results with different parameters, as in the case of full latency and dual latency, when the error of one case may be a lot higher or lower than that of the other case. Secondly, mathematical analysis sometimes has assumptions that are not valid for real world situations. For instance, assumptions of Poisson arrivals, independent service time, etc. may not be good approximations to real world situations. Small changes in the operation may have dramatic impact on a mathematical model. But a simulation program can be modified relatively easily.

A. Simulation Language

Two discrete event simulation languages were considered for use: GPSS and SIMSCRIPT. GPSS was chosen because of its compact structure that allows minimal changes in source codes when simulating different system parameters. The simulation program was originally developed in GPSS/PC on an IBM PC. The GPSS/PC compiler was very user friendly and had a lot of visual aids for debugging programs. But later we found that it was too slow to run the simulation program, so we switched to GPSS/H compiler[10] on an Amdahl 5860 main frame. The program running on the main frame was a few hundred times faster than the PC version. Furthermore, the GPSS/H compiler has additional facilities and block statements to reduce the number of source codes required and is upward compatible with GPSS V. For a description of GPSS language, see Schriber[12] and Gorden[11].

Compared to GPSS/V and GPSS/360 compilers, the GPSS/H compiler is faster and easier to use. For the models we ran, the GPSS/H compiler was about 6 to 7 times faster than the GPSS/360. Further, the GPSS/H compiler provides convenient features such as free input formats and interactive debugging.

A pre-processor was developed in PASCAL to allow convenient entry of different system parameters. The pre-processor automatically generates source codes for the GPSS/H compiler. The system parameters include service discipline (ordinary, gated or exhaustive), token transmission operation (multiple-token, single-token or single-packet), latency types (full latency or dual latency), service time distributions (exponential, constant or mixed of both), arrival distributions (exponential or constant), and running method (regenerative method or independent replications of runs).

B. Simulation Models

The block diagrams of the important modules are listed in Figure 2 to 4. The standard GPSS block notations and the concise meaning of each block can be found in the references[11,12]. The modules used for different service disciplines will be explained first, followed by the modules used for different token operations and finally the avalanche simulation module and the turbo module. The block diagrams should give enough information to understand the basic operations without too much emphasis on the programming details. The avalanche simulation or turbo mode can be used on top of the other modules. For illustration purpose, the figures show the modules with regenerative method and limited-to-one service; the modules with independent replications of runs will be similar. The main difference is that for independent replications of runs, the data is collected in a number of runs; the number is determined by the user. Thus, the user has to look at the output and calculate the data manually. As will be shown later, the calculation is fairly straightforward.

1) Service Disciplines– Figure 2 depicts the module used for the limited-to-one service discipline. The module segment shown is just for one station and the dashed rectangle represents the portion for each station. A circle in the block diagrams represents a block location. The first block statement for each station is a GENERATE block to generate its own traffic based on some distributions. The software supports two interarrival distributions: exponential and constant. Other kinds of distributions can also be entered easily. After a message is generated, it will go to a QUEUE block to start to gather waiting timing statistics. The message (or transaction) is then put into the user chain of that station. That makes the message *scan-inactive* and saves computing time dramatically. A free token circulates around the ring in a cyclic order. When the token finds a message waiting in a station, it will release one message from the user chain of that station. The token is then destroyed by the TERMINATE block. The message released from the user chain then seizes the facility (or single server) named as "TOKEN". The message will depart the queue and record the statistics. The server will be used for a certain amount of time depending on the service time distribution.

The software supports three different service distributions: exponential, constant and a combination of exponential and constant distributions. Other distributions can be entered easily. Finally the server becomes free again in the RELEASE block. For a dual latency station, the time difference between the full latency and the dual latency has to be added as a delay. For the regenerative method, the model tests if the whole system is empty. If it is, the statistics will be recorded as a regenerative cycle before the system will be reset. For the independent replications of runs, the statistics collected for the first M messages specified by the user are discarded and then the data afterward will be used as useful data.

In a gated service system, at the instant the token detected the station has message(s) to send, it stores the number of messages in the queue to a variable. Therefore, every time the server serves a message, the variable is decreased by one. The server will continue to serve until the becomes zero. In an exhaustive service system, the server will continue to serve until the queue of the station becomes empty. Thus, one would expect a longer running time for gated service compared with exhaustive service, assuming everything else the same, because there are more block statements involved in running a program for gated service.

2) Token-Operations– For the cases of the single-token and the single-packet operations, the situations are more complicated than the multiple-token and require more computing effort. Figures 3a, 3b and 3c show the block diagram of a dual latency ring using single-token operation. For the purpose of illustration, limited-to-one service and regenerative method are assumed. Using the full latency method, the ring latency is known. (The ring latency is equal to the sum of all station latencies plus propagation delays.) But the ring latency of a dual latency ring is not known beforehand and is dependent on the state of the ring. Therefore, using the dual latency method for the single-token and the single-packet requires special attention.

For the single-token operation, if the full latency method is used, the operations of the blocks become very similar to the multiple-token. The only difference is that the service time for the last message will be compared with the sum of the ring latency and the header time. If the service time is larger, the message will proceed as usual; otherwise, the time difference between the two has to added as additional delay induced. For ordinary service, the last message served per token is also the only message served per token.

If the dual latency method is used, the station will generate messages as normal. But after a message seizes the token and goes through a DEPART block, another transaction has to be generated for the dual latency method and it is called the *created transaction*. The created transaction is generated by the SPLIT block. The created transaction is necessary in order to check the time it takes for the header to go back to the same station. Since for the dual latency method the ring latency is not known beforehand, the created transaction is to be circulated around the ring. The ring latency experienced by the created transaction depends on the queues of other stations. After the created transaction goes back to its sending station, we check if the logic switch "RING1" is set. If the logic switch "RING1" is not set, it will be held in the GATE block waiting for the logic switch to be set by the original transaction. It the logic switch is set, the created transaction will proceed to set the logic switch "RING2" before finally being destroyed.

The original transaction will go through the normal path to the ADVANCE block and then the RELEASE block. In order to keep the memory for the system manageable by the GPSS/H compiler, two logic switches "RING" and "RING2" are used to control the flow of the two transactions. After the RELEASE block, the original transaction will set the logic switch "RING1" and then test if the created transaction has returned to its own station. If the created transaction has returned, the message transaction will proceed; otherwise, it will wait for the created transaction to return to the LOGIC block to set the logic switch "RING2". The setting of the logic switch "RING2" is controlled only by the created transaction and the setting of the logic switch "RING1" is controlled only by the original transaction. The reason for using the logic switches is to avoid using the TEST block alone, which produces heavy computing overhead.

The difficult part is when the station uses service disciplines other than ordinary service. For instance, if the station uses exhaustive service, it does not know how many messages that will be served in advance. That

is the main reason for using the created transaction. Only the original transaction of the last message for a token will open the logic switch "RING1" and allows it to check if *all* the created transactions have returned to the sending station. The operations described above assume that a station can switch from low latency to high latency between messages in the cases of exhaustive or gated service disciplines.

For the single-packet operation, if the full latency is used, the operations of the blocks become very similar to the multiple-token operation. The main difference is that the service time of the last message will be increased by the amount equal to the ring latency. The ring latency is a constant and can be calculated in the pre-processor. The message transaction will proceed as in the case of the single-token operation until it reaches the RELEASE block. For the single-packet, the entire message has to be received by the sending station rather than only the header. Therefore, we make use of the transit time to calculate the ring latency of the last message (as for exhaustive service). After the last message is served, we save the absolute clock using the MARK block. The transit time for each created transaction is stored in the same variable, so that the variable will finally store the transit time of the last created message after all created transactions are destroyed. In the mean time, the last message may be waiting for the last created transaction to finish. Therefore, the additional delay is equal to the transit time of the last created transaction minus the time the last message waited for all created transactions to finish.

3) Avalanche Simulation– This module tries to simulate the real-time behavior when the ring system experiences emergency messages arrived during a short period[15]. Although the avalanche arrivals may be large, the regular arrivals may be fairly small. There are two measures proposed: *relaxation time* and *empty time*. Relaxation time here is defined as the time it takes the system to finish serving all the avalanche messages.

Empty time is defined as the time it takes the system to empty the queues of all stations. Obviously, the empty time is larger than or equal to the relaxation time. The module requires the use of groups in GPSS to distinguish between avalanche messages and regular messages.

4) Turbo Mode– The model segment for the *turbo mode* is shown in Figure 4 and is important to saving computing time. Normally, without the turbo mode, the computing cost increases when the system utilization goes down. It is a common problem for discrete-event simulation because every time the simulated time moves forward, it is considered by the computer as an event and there is some computing overhead for every event. Hence, for the token ring LAN, the transfer of a free token increases the simulated time and thus is regarded as one event. But that is not useful to the data we are interested in. As the channel utilization becomes smaller, a larger portion of computing time is wasted in the transfer of the free token around the ring. In order to alleviate the problem, when the system becomes empty, the token is held in the TEST block. The scanning algorithm of GPSS then schedules the next event which will be the earliest message arrival. After a message arrival, the free token is put back to the ring randomly allowing messages a chance to capture it. It should be noted that the message first arrived does not guarantee the capture of the token, since another message may arrive later but could seize the token on the way the token goes to the station of the first message. There is a BUFFER block has to added to the module segment for each station. The BUFFER block should be put right after the QUEUE block and is used to explicitly force a restart of the current events chain scan. Otherwise, after a message is put into the user chain the compiler may misinterpret that there is no more active transaction and quit running.

C. Statistical Considerations

Computer simulation is analogous to doing random experiments; statistical analysis is frequently used. In this section, we are primarily interested in steady state statistics rather than transient state statistics. There are problems associated with simulation of steady state behavior. The first common problem for steady state simulation is that the duration of a transient phase is usually unknown. Therefore, if the transient phase duration is overestimated, then computing time is wasted. If the duration is underestimated, the data collected is biased because of the transient data included. A general technique to determine the duration of a transient is not known. In practice, an experimenter could estimate the duration by looking at the plot of the sequence and subjectively determine that a certain period is good enough to terminate a transient phase. After the transient duration is estimated, there remains the problem of analyzing the output data. The difficulty lies in the fact that the output sequence usually is not independent. The two approaches described below both produce independent blocks of data so classical statistics can be applied.

1) Independent Replications of Runs–

The most common and easiest to understand approach is to run the simulation using the method of independent replications. In order to make each replication independent of each other, we can change the seeds of random number generators for each replication, discard the data from the transient phase and collect the data afterward for each replication.

Suppose we want to estimate the true mean μ and the confidence intervals of the estimated mean $\hat{\mu}$. Let M be the number of replications, n be the sample size of each run, y_{ij} be the ith observation in the j run, the sample mean of the jth run be $\overline{Y}_j$, $s_{\overline{Y}}$ be the standard deviation of $\overline{Y}_j$ and $t_{M-1}^{\alpha/2}$ is the upper $\alpha/2$ percentile of the Student's t-statistics with (M-1) degree of

freedom. Hence, traditional statistical analysis can be used to calculate the confidence intervals. The $(1-\alpha)$ confidence interval may be obtained from the following expression:

$$\hat{\mu} \pm \frac{t_{M-1}^{\alpha/2} S_{\bar{Y}}}{\sqrt{M}}$$

In general, M should be kept small and N large to minimize unused data during the transient state and any residual bias caused by the slow convergence of μ.

2) The Regenerative Method– The regenerative method[13,14] is based on the idea of independent cycles in a single run. The advantages of using the regenerative method are listed as follows:

1. It does not need to know the duration of a transient phase.
2. It collects useful data right at the beginning. We don't need to discard any useful data in the transient phase.
3. It produces independent sub-sequences of data in a single run.

The regenerative method applies to discrete-event simulations that can be modeled as regenerative processes. The basic regenerative assumption is that there exists a regeneration time, T_1, such that the continuation of the process beyond T_1 is a probabilistic replica of the whole process starting at time 0. This property implies that $T_2, T_3, \ldots$ exist and they have the same property as T_1. Further, finite regeneration times are assumed. The typical situation which satisfies the regenerative requirement is when the system hits T_i, the system can proceed without any knowledge of its past history. The procedures of obtaining an approximate $(1-\alpha)$ confidence interval for the parameter r can be found in the references[13,14].

D. Validation

The validity of the simulation models was first inspected by using the interactive debugging facilities in GPSS/H. The program was monitored to determine if it behaved as expected. Afterward, the validation was then checked by comparing known results with the simulation results. The two examples of Ferguson's paper[7] were chosen for comparison and the simulation results with 90% confidence intervals covered most of the data points of the known theoretical results.

When the simulations for unknown theoretical results were run, there were three things that had to be checked. Firstly, we would observe if the results were too unreasonable. If they were, the systems parameters would be inspected again very carefully. Secondly, when the utilization went down to 0.1, we know that the mean message waiting time should be close to but not smaller than half of the ring latency for Poisson arrivals. Thirdly, the mean message waiting time of the full latency simulations should be more than that of the dual latency simulations for the same utilization. Finally, when the utilization went up close to the stability limit, the simulation results should experience a sharp increase in mean message waiting time. The simulation should be stopped because the memory requirements and the computing cost would rise dramatically. The simulation might become too expensive to run.

E. Features of the Simulation Models

The simulation model consists of a pre-processor, a processor and a post-processor. The pre-processor was written in PASCAL and it automatically generates the source code for the GPSS/H processor after the system parameters are entered. If the regenerative method is specified, a

FORTRAN post-processor is available for calculation and output of confidence intervals for the mean message waiting time and other statistical data such as the number of regenerative cycles, number of messages, etc.

The pre-processor is the heart of the simulation software. It allows the user to enter his parameters for his ring system relatively easily. Also, it checks if the input parameters are unrealistic. If they are, it will output a warning message, but a GPSS/H program will still be generated. This will prevent the user from wasting computing money on wrong entries and at the same time give the user the freedom to run such a system. The post-processor is a relatively straightforward program for statistical analysis of the regenerative method.

It should be noted that the simulation model can be used for either a *token ring* or *token bus* network. Of course, the input parameters will be different. For a token ring, the switchover time is just the propagation time from one station to the next plus the station latency. For token bus, the switchover time would become the time to transmit an explicit token frame, which is 152 bits in the IEEE 802.4 Token Bus standard plus the propagation delay between the two stations, which may be physically quite far apart. Finally the response time of the station has to be added. In summary, the input parameters and options are listed as follows:

PARAMETERS	OPTIONS
Service Disciplines	exhaustive, gated, or limited-to-one.
Token Transmission Types	multiple-token (e.g. FDDI, IEEE 802.4), single-token (e.g. IEEE 802.5) or single-packet.
Latency Types	full latency or dual latency.
Interarrival Time	exponential, constant or a given distribution function.
Service Time	exponential, constant, a mix of exponential and constant, or a given distribution function.
Switchover Time	constant for both latencies or a given distribution function.
Number of Stations	2 to 100.
Running Methods	regenerative method or replicated runs.
Turbo Mode	yes or no for running low utilization.
Avalanche Simulation	yes or no.

V. SIMULATION RESULTS

This section presents some of the simulation results in order to illustrate the usefulness of the simulation model. Since there can be so many combinations of the parameters, it was difficult to choose a so called "typical" LAN to investigate. Thus, we used the values for the low latency and the full latency based on the actual implementation[3]. The values used for the latencies were based on the IEEE 802.5 implementation that runs at 10 Mbps data rate. The station low latency is 0.6 bits and the station full latency is 3.6 bits.

It is assumed that the LAN has 20 stations, 2 of them heavily loaded and 18 of them lightly loaded. This asymmetric loading was studied before[7] and was chosen to represent a class of asymmetric traffic. Each of the two heavily loaded stations contributes 40% of the total utilization while each of the eighteen lightly loaded stations shares the remaining traffic utilization evenly.

Case 1 was chosen to illustrate the effect of message lengths on dual latency while case 2 was chosen to illustrate the effect of message length distributions from totally exponential gradually to purely constant. In case 1, we used the exponential part to be 512 bits and then decreased it by one half to 256 bits and so on to 32 bits. But we kept the overhead the same (48 bits). Figure 5 shows the performance improvement expected for case 1 with different mean message lengths. Note that only the mean waiting time for station 1 is shown. Stations 1 and 8 are heavily loaded and of primary concern. The behavior of the mean waiting time for station 1 and station 8 are very similar. From the figure, the first observation is that the improvement increases rapidly as the utilization goes up. Secondly, when the message length increases the advantage of using the dual latency method decreases. It is due to the fact that long messages have a lower switchover time overhead than short messages. Case 2 tries to investigate

what would happen when the mean message length is dominated by either the constant field or the exponential field. Figure 6 shows that the performance improvement becomes more significant as the mean message length has more constant field.

In real systems like the IEEE 802.5 and the FDDI standards, they usually have a constant header field. That will make the dual latency more appealing than random message lengths. For real-time applications, the system often has a lot of short and constant messages that are time-critical. That makes the dual latency stations even more attractive than the full latency or regular stations.

VI. CONCLUSION

The objective of this research was to develop a simulation model that is flexible enough to simulate most token passing networks. Specifically, we determined the performance improvement of using the dual latency method. The fact that we can change many system parameters gives us a great deal of flexibility to examine a wide range of applications. It is conceived that this simulation tool not only can be useful for network design but also for fine tuning of network operation. The simulation provides important data for design trade-offs. Often the simulation identifies bottlenecks in the design and suggests alternatives that can reduce cost and/or improve performance.

The study shows that a dual latency ring always performs better than a full latency ring. The degree of improvement depends on a number of parameters: number of stations, message length statistics, and utilization of the system. The improvement in mean message waiting time is more significant when the LAN has a large number of stations, short messages and/or constant messages.

VII. SUGGESTIONS FOR FUTURE WORK

A. *Real-Time Applications*

Many existing LANs have been developed for the office environment. However, there is a demand of LANs for real-time situations. It is unclear if the currently available protocols would be suitable to handle real-time applications. The importance of modeling of avalanche traffic (e.g. alarm messages) for real-time systems was stressed[15]. Since the dual latency method is more suitable for real-time applications, avalanche simulation may be able to reveal other important properties of the dual latency method. Hence, modeling of avalanche simulation should be further investigated.

B. *End-to-end Performance Modeling*

The methods available for performance modeling of the physical layer, and MAC layer are quite successful in general and proven useful in practice. Nevertheless, the performance of the first two layers may not be the performance experienced by a user. That makes the end-to-end performance very useful; however, universal techniques for end-to-end performance modeling are not yet available[16]. The end-to-end performance modeling may involve analysis of a network of queues. Since the simulation software was developed for the first two layers, higher layer models may be developed and added on top of the software to predict the end-to-end performance.

Acknowledgement: Special thanks are due to I.R. Radziejewski, who implemented the dual latency method network interface unit in hardware and has helped us in many aspects throughout the course of this research. We also would like to thank Dr. D.A. George, the Dean of Applied Science of Simon Fraser University for his thoughtful comments and encouragement. This project is partially supported by the Natural Sciences and Engineering Research Council of Canada and the Center for Systems Science of Simon Fraser University.

REFERENCES

[1] M. Schwartz (1977), *Computer Communication Network Design and Analysis*, Prentice-Hall, 1977, p. 275.

[2] W. Bux (1981), "Local area subnetworks: a performance comparison", *IEEE Trans. Commun.*, vol. COM-29, pp. 1465-1473, October, 1981.

[3] R.H.S. Hardy, Edward Lo and Ian Radziejewski (1987), "Utilization of Dual Latency Stations for Performance Improvement of Token Ring Networks", *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing*, June 4-5, 1987, Victoria, Canada, pp. 395-399.

[4] R.H.S. Hardy, Edward Lo and Ian Radziejewski (1988), "Performance Evaluation of a Token Ring Network with Dual Latency Stations", *IEEE INFOCOM '88*, Mar. 27-Apr. 1, New Orleans, U.S.A, 1988, pp. 487-495.

[5] H. Takagi (1986), *Analysis of Polling Systems*, MIT Press, 1986.

[6] H. Takagi (1987), "A survey of queueing analysis of polling systems", *Proceedings of the Third International Conference on Data Communication Systems and Their Performance*, Rio De Janeiro, Brazil, June 22-25, 1987.

[7] M.J. Ferguson and Y.J. Aminetzah (1985), "Exact results for non-symmetric token ring systems", *IEEE Trans. Commun.*, vol. COM-33, pp. 223-231, 1985.

[8] W.P. Groenendijk (1987), "Waiting-time approximations for cyclic-service systems with mixed service strategies", *Centre for Mathematics and Computer Science*, Amsterdam, Netherlands, 1987.

[9] M.J. Ferguson (1986), "Mean waiting time for a token with station dependent overheads", in R.L. Pickholtz (ed.), *Local Area & Multiple Access Networks*, Computer Science Press, pp. 43-67, 1986.

[10] James O. Henriksen and Robert C. Crain (1983), *GPSS/H USER'S MANUAL*, Wolverine Software Corporation, 1983.

[11] Thomas J. Schriber (1974), *Simulation Using GPSS*, John Wiley & Sons, 1974.

[12] Geoffrey Gorden (1975), *The Application of GPSS V to Discrete System Simulation*, Prentice-Hall Inc., 1975.

[13] M.A. Crane and D.L. Iglehart (1975), "Simulating Stable Stochastic Systems, III: Regenerative Processes and Discrete-Event Simulations", *Operation Research*, vol. 23, no. 1, pp. 33-45, January-February 1975.

[14] M.A. Crane and A.J. Lemoine (1977), "An Introduction to the Regenerative Method for Simulation Analysis", *Lecture Notes in Control and Information Sciences*, Springer-Verlag, New York, 1977.

[15] G. Le Lann (1985), "Real-Time Protocols", Ch. 14, *Local Area Networks: An Advanced Course* (D. Hutchison, J.A. Mariani and W.D. Shepherd, Ed.), Springer-Verlag, New York, 1985.

[16] M. Reiser (1986), "Communication-System Models Embedded in the OSI-Reference Model, A Survey", *Computer Networking and Performance Evaluation*, Elsevier Science Publishers B.V., North-Holland, 1986.

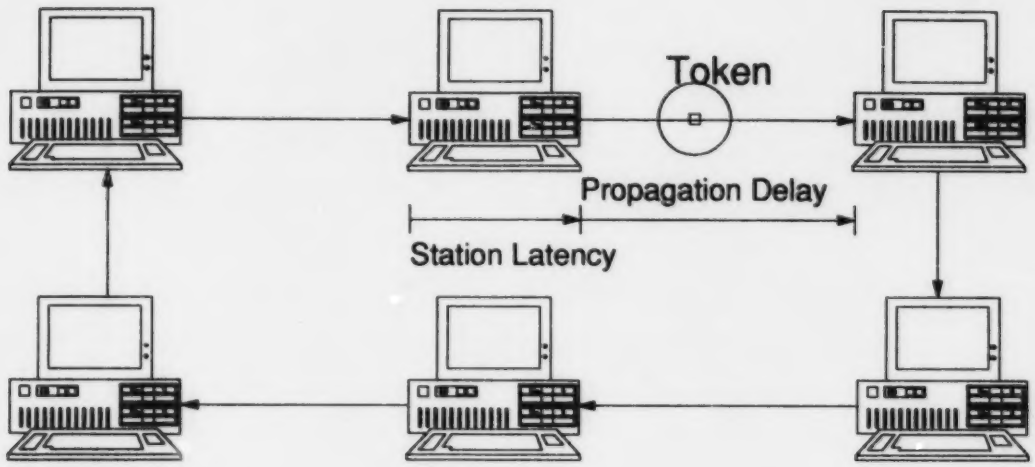

Figure 1 - A Token Ring Network with Six Stations Connected.

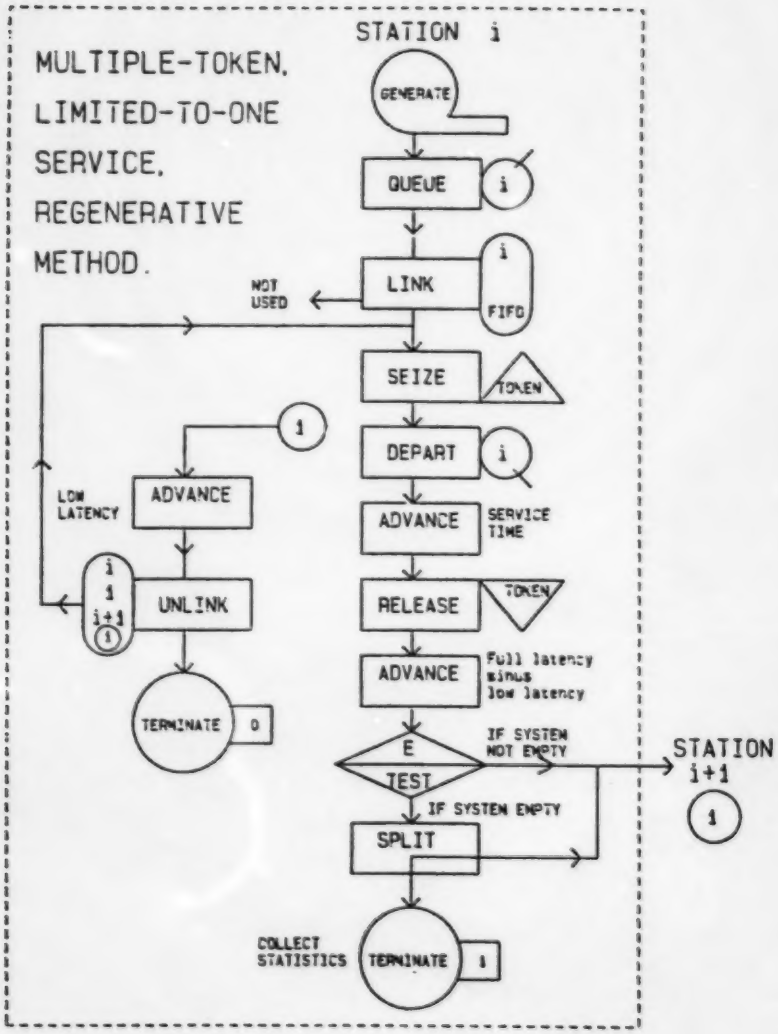

Figure 2 - Module Segment for Limited-to-one Service Discipline with Multiple-Token Operation and Regenerative Method.

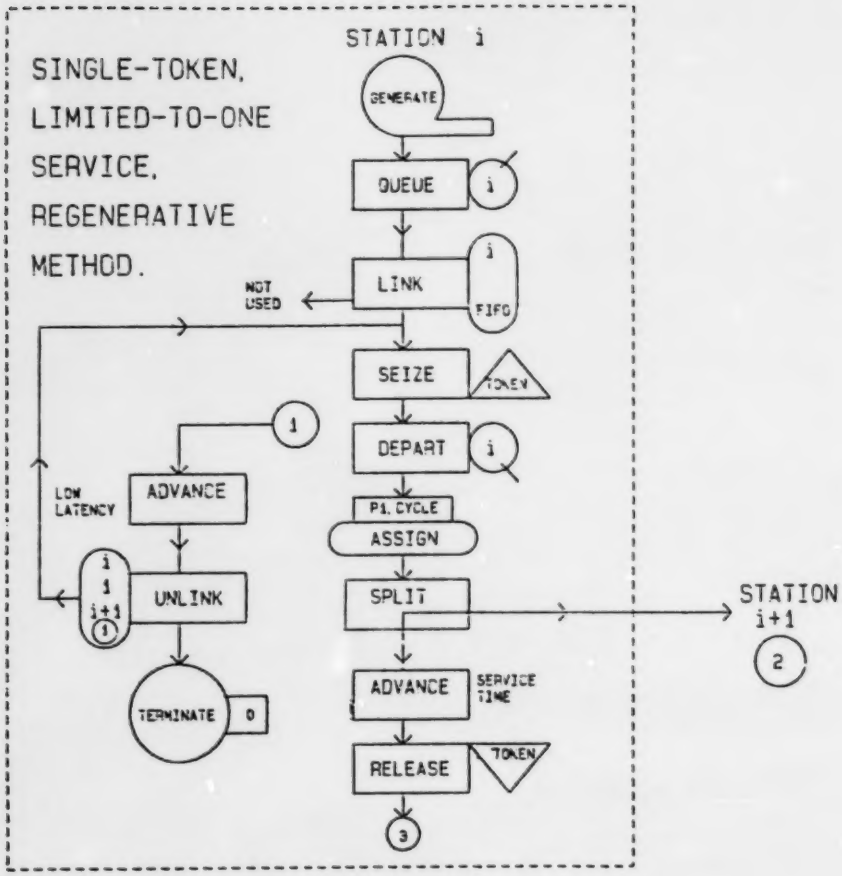

Figure 3a - Module Segment for Single-Token Operation with Limited-to-one Service Discipline and Regenerative Method.

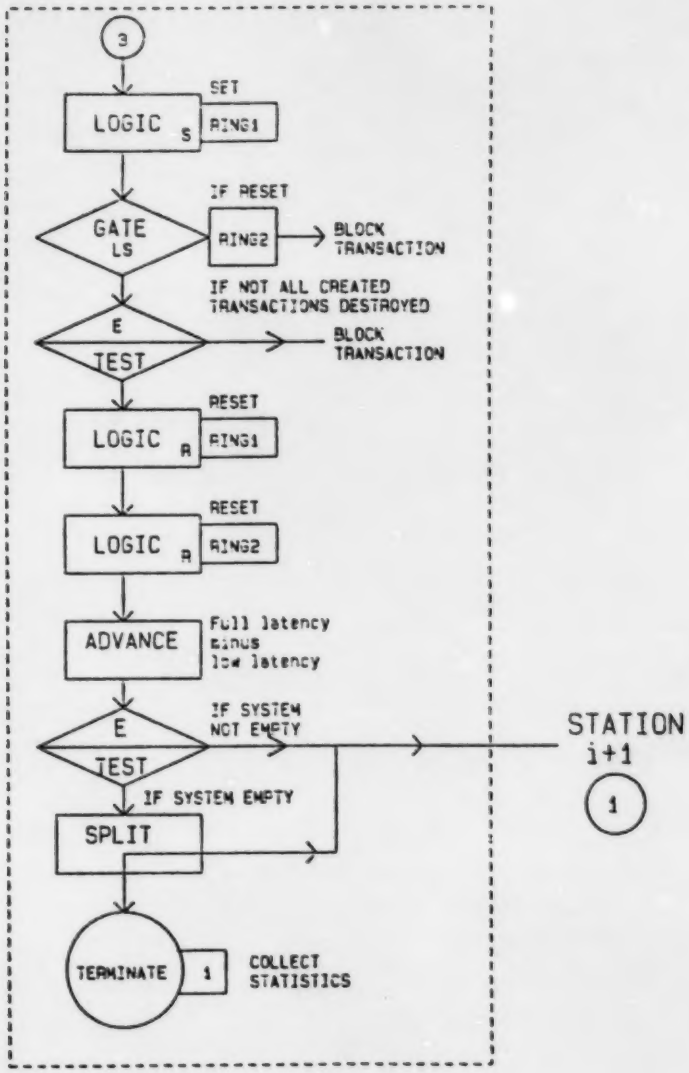

Figure 3b - Module Segment for Single-Token Operation with Limited-to-one Service Discipline and Regenerative Method.

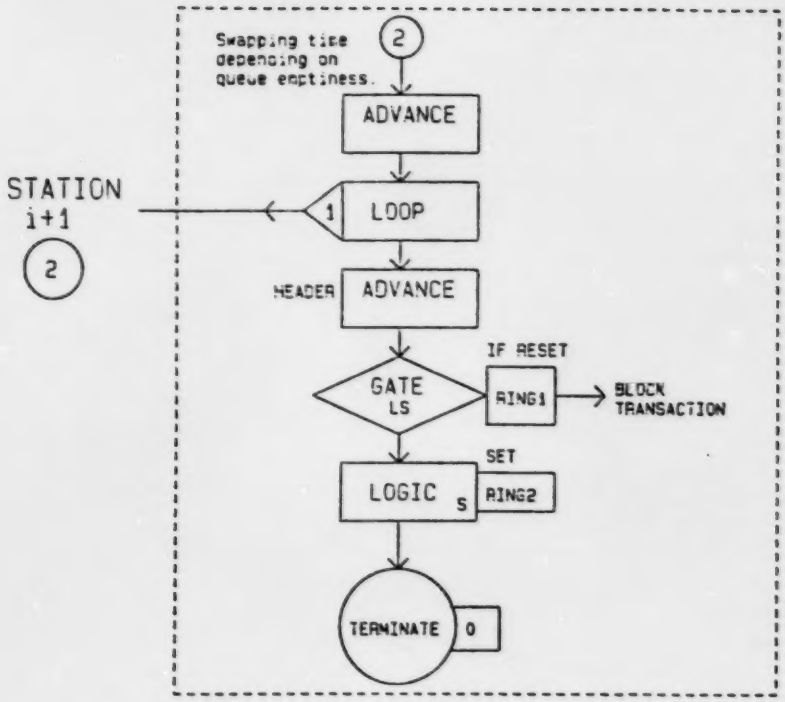

Figure 3c - Module Segment for Single-Token Operation with Limited-to-one Service Discipline and Regenerative Method.

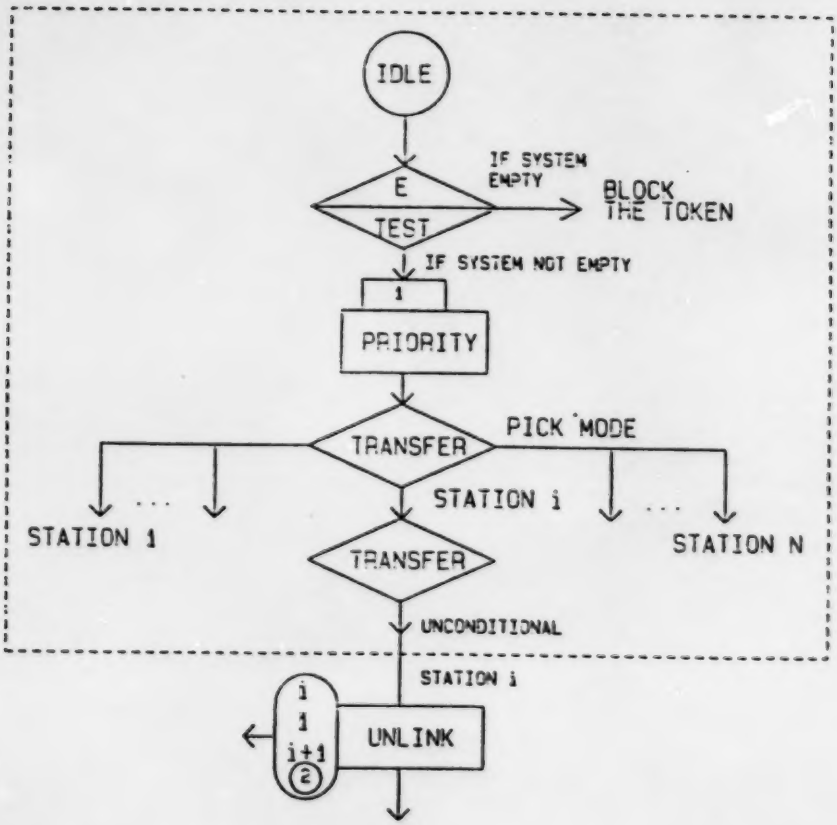

Figure 4 - Module Segment for the Turbo Mode.

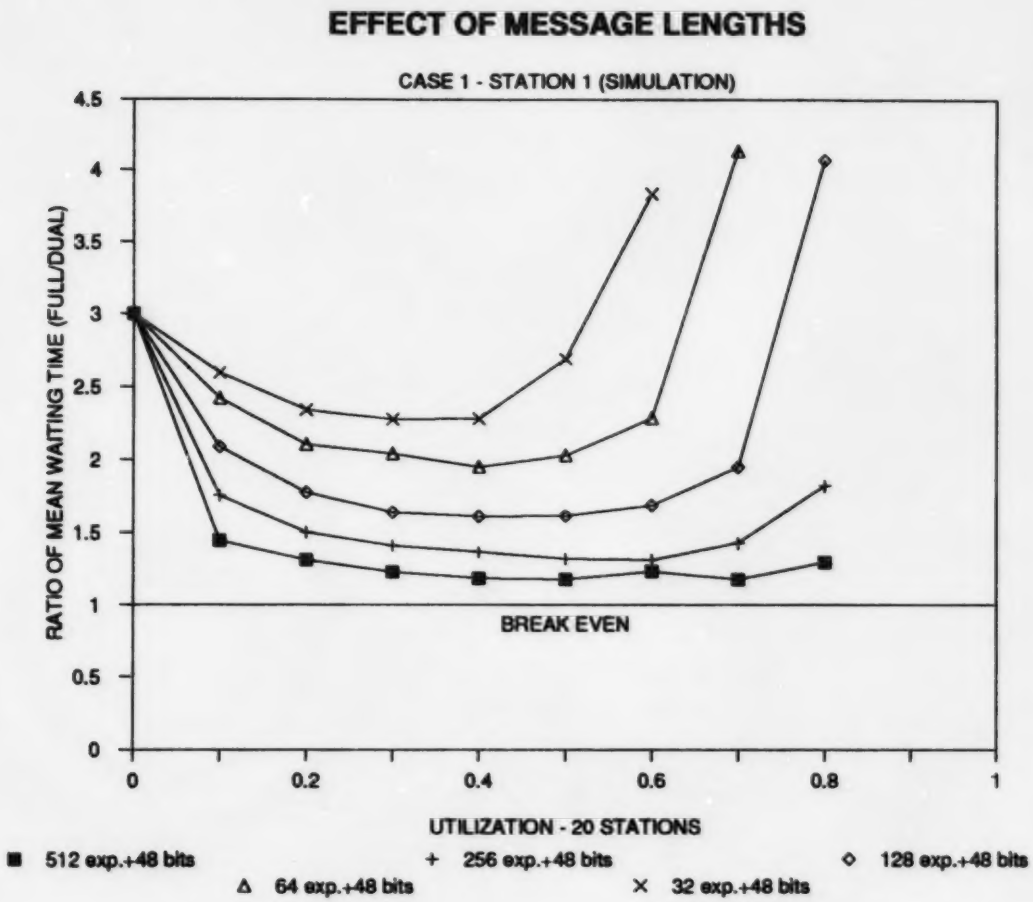

Figure 5 - Performance Improvement of Full Latency to Dual Latency for Case 1, different mean message lengths

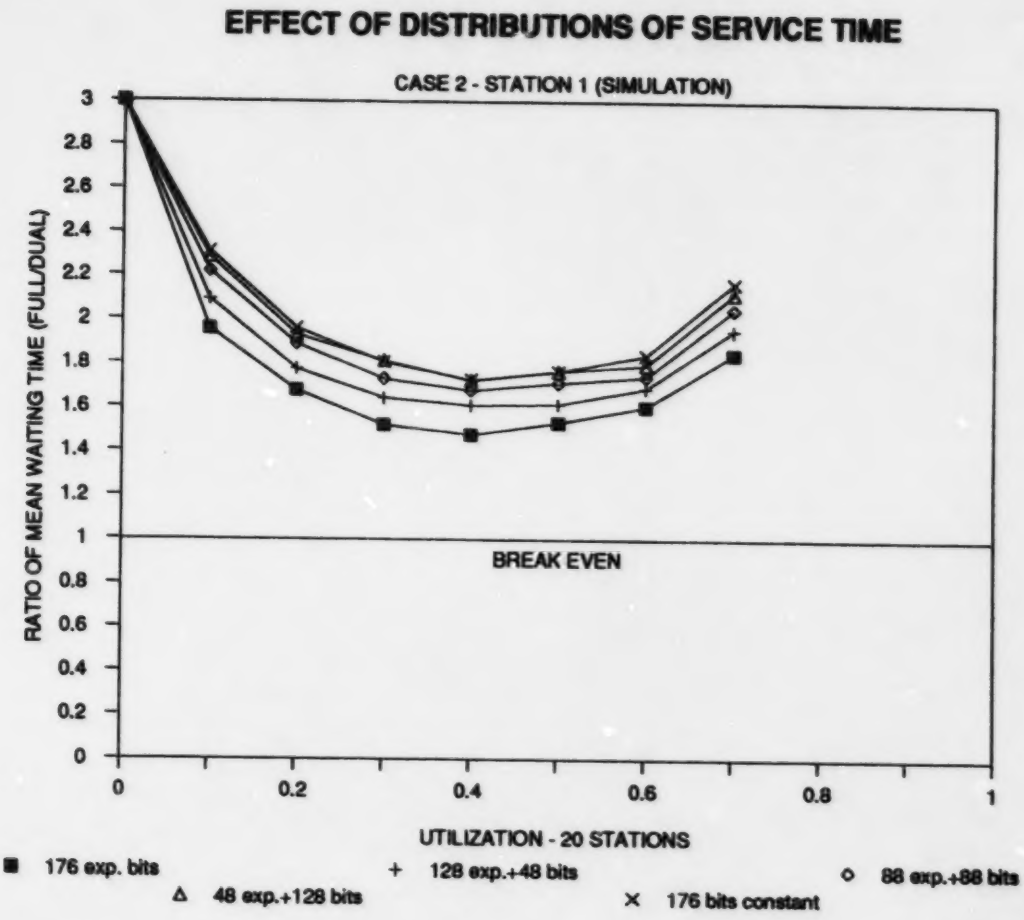

Figure 6 - Performance Improvement of Full Latency to Dual Latency for Case 2, different message distributions

Centre for Systems Science

• Simon Fraser University established the **Centre for Systems Science** as a provincial centre for advanced study. The objective of CSS is twofold: coordinate and combine advanced systems research at SFU, and encourage application of research results within industry in order to further the technological development of British Columbia.

•CSS has assembled three research groups to streamline its efforts. The orientation of these groups is as follows: intelligent systems, microelectronics, and computer and communications systems. Because of shared research interests, this latter group merges with LCCR (**Laboratory for Computer and Communications Research**) for the publication of technical reports.

• CSS's three groups involve over 80 skilled faculty researchers principally from SFU's Schools of Engineering Science and Computing Science, but also draw substantially on Mathematics, Physics, Geography, Philosophy, Linguistics, Psychology, Communication, and Kinesiology.